An Application–Level QoS Comparison of Inter–Destination Synchronization Schemes for Continuous Media Multicasting

Toshiro Nunome and Shuji Tasaka
Department of Computer Science and Engineering, Graduate School of Engineering, Nagoya Institute of Technology, Nagoya 466–8555, Japan

Abstract—This paper presents an application–level QoS comparison of three inter–destination synchronization schemes: the master–slave destination scheme, the synchronization maestro scheme, and the distributed control scheme. The inter–destination synchronization adjusts the output timing among destinations in a multicast group for live audio and video streaming over the Internet/intranets. We compare the application–level QoS of these schemes by simulation in which we assume an intranet. From the comparison, we clarify their features and find the best scheme in the environment. The simulation result shows that the best scheme depends on the network configurations and acceptable inter–destination synchronization quality of applications.

I. INTRODUCTION

Multicasting is an important technique for live audio and video streaming applications. Usually, multicasting in the networks employs IP multicast [1], which provides the best–effort service and no QoS (Quality of Service) control mechanism. Thus, the temporal relations of continuous media may be disturbed by delay and its jitter during the transmission.

This disturbance can be a serious problem for interactive and collaborative applications such as quiz show and distance learning. In a quiz show, for example, contestants may feel unfairness because the contestant at the shortest delay destination gets an advantage over the others. In order to preserve the temporal relations, we can exert media synchronization control [2], which is one of the application–level QoS control [3].

We can identify three types of media synchronization: intra–stream synchronization, inter–stream synchronization and inter–destination (or group) synchronization. The intra–stream synchronization control is necessary for the preservation of the timing relation between media units (MUs) such as video frames in a single media stream; an MU is the information unit for media synchronization. The inter–stream synchronization is required for keeping the temporal relations among MUs in multiple media streams. The inter–destination synchronization control is needed in multicast communications. The purpose of the control is to output each MU simultaneously at different destinations. In multimedia conferencing, for instance, if the output timing of speech by a participant largely varies from destination to destination, the conference itself cannot hold. Furthermore, as the size of the multicast group becomes large, the difference in the output timing between destinations also becomes large. Thus, inter–destination synchronization is an indispensable function to support these applications, especially in large multicast groups.

We can find several researches on inter–destination synchronization in the literature [4]–[10]. A flow synchronization protocol is proposed in [4], where an initiator manages the distribution of control information among destinations. It is also assumed that globally synchronized clocks are employed; that is, clock ticks at the sources and destinations have the same advancement, and the current local times are also the same. However, the validity of the protocol has not been demonstrated sufficiently. Akyildiz and Yen present group synchronization protocols in [5], where clocks run at different rates (i.e., locally available clocks). By simulation, they evaluate the maximum, minimum and average amount of asynchrony in seconds. However, the simulation assumes only a single media stream and employs a dummy stream as the media. Furthermore, they assume that the network delay bounds are known; however, the bounds cannot be known exactly in the Internet. In [6], Benslimane proposes an inter–destination synchronization scheme which does not need globally synchronized clocks. In the scheme, a source terminal manages control information and then announces it to all the destinations. He also assumes that the network delay bounds are known. The effectiveness of the proposed scheme has been shown by simulation; however, he does not consider the temporal structures of media streams.

The papers mentioned above have some limitations when we apply their schemes to continuous media transfer. For example, they assume that the network delay bounds are known and do not consider the temporal structures of continuous media. That is, they do not discuss media synchronization quality; we regard the quality as the major part of application–level QoS.

On the other hand, in [7]–[9], inter–destination synchronization schemes based on the virtual–time rendering (VTR) media synchronization algorithm [11] are proposed; they are the master–slave destination scheme, the synchronization maestro (or synchronization manager) scheme, and the distributed control scheme. The VTR algorithm is applicable to networks with unknown delay bounds by dynamically adjusting the MU rendering–time according to the network condition. It employs globally synchronized clocks.

The master–slave destination scheme is proposed in [7]. In this scheme, destinations are grouped into a master destination and slave destinations. Each slave destination adjusts the output timing of MUs to that of the master destination. This is suitable for applications in which a single destination has priority over the others; in multimedia conferencing, for instance, we can select the chairperson’s terminal as the master destination. However, the scheme cannot treat all the destinations fairly.

The synchronization maestro scheme is proposed in [8]. This scheme can handle all destinations fairly. In this scheme, the synchronization maestro collects output timing information from the destinations and distributes control information to them in order to arbitrate the output timing at each destination.

The master–slave destination and synchronization maestro schemes are centralized control ones. In these schemes, if the master or maestro cannot communicate with the other terminals owing to some trouble, no destination is able to carry out the inter–destination synchronization control. In order to solve the problem, the distributed control scheme is proposed in [9].

The effectiveness of the master–slave destination scheme, the synchronization maestro scheme, and the distributed control scheme in terms of the application–level QoS was examined by simple experiment with a source and two destinations. For example, in [10], Tasaka et al. examine the influence of handover on the application–level QoS including inter–destination synchronization quality in an integrated wired and wireless net-
work, where the synchronization maestro scheme is adopted.

As multicast communications become popular, its scale will grow. However, the effectiveness of the inter–destination synchronization control schemes in multicast environments with many destinations has not been clarified. Furthermore, no quantitative comparison among these schemes has been made, although the qualitative comparison is performed in [9].

In this paper, we compare the three schemes for inter–destination synchronization in terms of application–level QoS by simulation in which an intranet is assumed. Accordingly, we clarify their features and find the best scheme in the environment.

The rest of this paper is organized as follows. Section II describes principles of the inter–destination synchronization schemes. Section III illustrates a methodology for the application–level QoS assessment, including the network configuration, simulation method and QoS parameters. The simulation results are presented and discussed in Section IV.

II. INTER–DESTINATION SYNCHRONIZATION SCHEMES

As the basis of inter–destination synchronization control in this paper, we employ the scheme in [12], which is based on the VTR media synchronization algorithm and proposed as a synchronization maestro scheme. We also modify the control scheme so that it can realize the master–slave destination control or the distributed control. We further enhance the distributed control scheme so as to smooth traffic due to the control packets.

In this section, we first describe an outline of the VTR media synchronization algorithm. We then explain the three inter–destination synchronization schemes: the synchronization maestro scheme, the master–slave destination scheme, and the distributed control scheme. Furthermore, we present the enhancement of the distributed control scheme.

A. VTR Media Synchronization Algorithm

The VTR algorithm selects a media stream as the master stream and the others as slave streams, which are synchronized to the master. The algorithm exerts intra–stream synchronization control over both master and slave streams, while it performs inter–stream synchronization control only on slave streams after the intra–stream control. In this paper, we consider the transmission of an audio stream and the corresponding video stream. Audio is selected as the master stream and video as the slave stream since audio is more sensitive to intra–stream synchronization error than video.

We first consider intra–stream synchronization control. The disturbance of media synchronization appears in some form of delay jitter; therefore, we can achieve media synchronization by absorbing the jitter at the destination. This is carried out by buffering MUs for an appropriate period of time. It is clear that the period of time should be the maximum delay jitter. However, we cannot necessarily set the buffering time to this value, because getting the exact value in the Internet is very hard, and even if we can know it, setting the value may destroy the real–time property.

The VTR algorithm assumes no exact knowledge of the network delay jitter; by utilizing the timestamp provided to each MU at the source, it adaptively changes the buffering time according to the amount of delay jitter of MUs received at the destination. Initially, the buffering time is set to a rough estimate of the maximum delay jitter, which is denoted by $J_{\text{max}}$; this value may be different from destination to destination. When inter–destination synchronization control is applied, however, a constant delay value $\delta$ instead of individual buffering times $J_{\text{max}}$’s is used commonly to all the destinations; this is referred to as the target delay time, which is defined as the time from the moment an MU is generated until the instant the MU should be output. After the first MU is received, the buffering time or the target delay time can be changed by the modification of the target output time of each received MU. The target output time is the time when an MU should be output. When the MU arrives at the destination too late after the target output time, the target output time itself is expanded to absorb the jitter. In order to preserve the real–time property of live media, we can set the maximum allowable delay $\Delta_{\text{max}}$ so that the modification of the target output time does not make MU delay exceed this limit. Furthermore, the target output time can be contracted when the amount of delay jitter decreases; this means that the buffering time decreases. Only the master stream can modify the target output time for itself, and accordingly the slave stream modifies it by the same amount at the same time.

Inter–stream synchronization control is exerted over the slave stream; the output timing of each slave MU is controlled so that the difference in output time between the slave MU and the corresponding master MU can agree with the difference in timestamp between the two MUs. In this paper, we suppose loosely–coupled media streams, where each slave MU is not provided with the sequence number of the corresponding master MU.

Inter–destination synchronization is achieved by adjusting the MU buffering time at each destination so that its output timing can be the same at all the destinations. We describe the three inter–destination synchronization schemes below.

B. Synchronization Maestro Scheme

The synchronization maestro scheme employs a synchronization maestro, which gathers the information on the output timing from all destinations and adjusts the output timing among the destinations by distributing control packets. The maestro can be chosen from among the sources and destinations.

At the beginning of the output of the first MU in the master stream, every destination inquires of the synchronization maestro whether the target output time should be modified or not, by sending the information on the output timing to the maestro. The purpose is to adjust the output timing of the succeeding MUs among all the destinations. In this paper, we represent the output timing in terms of the total slide time; it denotes the total amount of modification of the target output time. Therefore, the destination sends a recommended value of the total slide time to the maestro; it is referred to as the recommended total slide time in this paper.

At the beginning of the output, when the destination receives a constant number of consecutive MUs each of which has arrived later or earlier than its target output time ($N_c$ consecutive MUs have arrived later or $N_d$ successive MUs have arrived earlier [12]), it notifies the maestro of the recommended total slide time. The recommended total slide time is different from the total slide time in that the latter is the accumulation of the slide times, while the former is employed for inquiry about the modification of the target output time in advance. The destination also notifies the synchronization maestro of the total slide time whenever the target output time is changed by the intra–stream synchronization control. In order to decide the total slide time or recommended one, the scheme specifies three parameters: $r_1$, $r_2$ and $r_3$ [12].

When the synchronization maestro receives the total slide time or recommended one from each destination, it determines the reference value of the total slide time as the reference output timing. This is performed by comparing the output timings received from the destinations. Then, the maestro multicasts the information of the reference total slide time to all the destinations when the time is changed. It also multicasts the information at regular intervals (say, 5 seconds).

Each destination gradually adjusts its own total slide time to the reference one when it receives the information on the reference output timing. Parameters $r_4$ and $r_5$ [12] are defined in order to adding and subtracting the total slide time, respectively.
C. Master–Slave Destination Scheme

In the master–slave destination scheme, destinations are grouped into a master destination and slave destinations. Each slave destination does not send any information on the output timing. It adjusts the target output time of MUs to that of the master destination.

This scheme uses the total slide time or the recommended total slide time at the master destination as the reference total slide time of all the destinations. When the reference output timing is changed, the master destination sends it to all the slave destinations. In addition, the master destination periodically sends the total slide time for recovering loss of control packets (every 5 seconds in our simulation).

Each slave destination gradually adjusts its own total slide time to the reference one received from the master destination. It is performed in the same way as that for a destination under the synchronization maestro scheme. However, it should be noted that no slave destination sends any control packet including the information on the output timing.

D. Distributed Control Scheme

The distributed control scheme can perform inter–destination synchronization without the centralized control terminals such as the synchronization maestro and master destination. In the distributed control scheme, each destination decides the reference output timing from among the output timing of itself and that of the other destinations.

Each destination multicasts the total slide time or recommended one, which is decided in the same way as that of the synchronization maestro scheme, to all the other destinations. In addition, each destination periodically sends the total slide time as the master destination under the master–slave destination scheme does.

In this scheme, each destination also has the same function as the synchronization maestro. When the destination outputs an MU, it decides the reference total slide time from among the output timing of itself and that of the other destinations. Then, the destination gradually adjusts its own total slide time to the reference one in the same way as that in the two centralized control schemes.

E. Enhancement of Distributed Control Scheme

In the original version of the distributed control scheme, a control packet is generated just after the output of a voice MU. Thus, multiple destinations may send control packets at the same time because of inter–destination synchronization. As a result, bursty traffic due to the control packets degrades the output quality of media streams.

A variety of studies on reliable multicast protocols with retransmission–based error recovery have been reported [13]. These studies have solved the feedback–implosion problem [14]. That is, if every receiver reports the success or failure of the data transfer, the sender will be overwhelmed with feedback packets. As an example of the solution, each destination sets a random backoff timer before sending a feedback packet [15].

This paper employs the above approach for the distributed control scheme. That is, each destination sets a random backoff timer before sending a control packet. This timer generates a value uniformly distributed between 0 ms and 50 ms in units of 1 ms\(^1\). The destination sends the control packet after the waiting time generated by the timer.

\(^{1}\)Each control packet is generated just after the output of an voice MU. Thus, the minimum generation interval between two control packets equals the output interval between two MUs. Furthermore, the media synchronization algorithm in this paper works in the millisecond unit.

III. METHODOLOGY FOR QUALITY ASSESSMENT

We compare the application–level QoS of the inter–destination synchronization schemes by computer simulation with ns–2 (network simulator version 2) [16].

A. Network Configuration

Figure 1 illustrates the network configuration in the simulation. This is an example of an intranet. In this paper, we suppose \(2x − 1\) terminals receiving multicast live media; \(x\) denotes an arbitrary integer value and \(x \geq 3\). We can vary the scale of the network by changing \(x\).

\(R_k (k = 1, 2, \cdots, 2x)\) denotes a router node. MS is the source terminal node, and terminal \(MR_l (l = 1, 2, \cdots, 2x – 1)\) is a destination. Furthermore, \(LS_m (m = 1, 2, \cdots, 2x – 2)\) represents a load sender terminal, while \(LR_m\) is the load receiver terminal. Each connection between two routers is assumed as a serial line, which is a 4 Mbps duplex link with a transmission delay\(^2\) of 1 ms. The link between a router and a terminal is an Ethernet; the transmission rate of the link is 10 Mbps, and the transmission delay is 0.1 ms. Each link has a first–in first–out (FIFO) queue for each direction.

B. Method of Simulation

We assume MS as the video and voice sources. MS multicasts the media streams to all the destinations by using the RTP/UDP. We use a voice stream of ITU–T G.711 μ–law and an MPEG1 video stream. Table I shows the specifications of the voice and video. Furthermore, we take the media capturing and encoding delay time into consideration in the simulation. The capture duration of a voice MU equals the inter–MU time, which is 50 ms in this paper, and the encoding time is negligible; therefore, we set the capturing and encoding delay time of each voice MU to 50 ms. On the other hand, the capture duration of a video MU is just a moment. However, it spends much time to encode a video frame. In this paper, we set the capturing and encoding delay time of each video MU to 8 ms, which is the

\(^{2}\)The original ns–2 does not take the processing delay in routers into account. Thus, we set the transmission delay so as to include it.
same time as that in the experimental system in [17]. Each MU leaves the source capturing and encoding delay time after its timestamp.

In the simulation, we compare the application–level QoS of five schemes: NC (No Control), VTR, Maestro, Master–Slave, and Distributed. NC means that no media synchronization control is carried out. VTR exerts intra–stream and inter–stream synchronization control based on the VTR algorithm. That is, it does not employ any inter–destination synchronization mechanism. Maestro and Master–Slave denote the synchronization maestro scheme and the master–slave destination scheme, respectively. Distributed means the distributed control scheme with the random backoff timer.

In the VTR algorithm, we set the target delay time δ and the maximum allowable delay Δt1 to 100 ms and 300 ms, respectively. In addition, we set Nc = 10, Nj = 20, r1 = ∞ and r2 = r3 = r4 = r5 = 20 ms. The other thresholds and parameters in the VTR algorithm have the same values as those in [17]. Furthermore, the synchronization maestro scheme and the distributed control scheme select the latest output timing among the collected output timings as the reference 4.

LSm and Lrm are used to handle traffic flows of interference. The load sender terminal LSm sends fixed–size IP datagrams of 1500 bytes each to the related load receiver terminal Lrm at exponentially distributed intervals. The amount of the interference traffic is adjusted by changing the average of the interval. The interference data stream from LSp (p = 1, 3, ..., 2x − 3) to LRp has the same amount of average load for all pairs; we refer to it as average load 1. LSq (q = 2, 4, ..., 2x − 2) also sends the same average load; it is called average load 2.

C. QoS Parameters

In order to assess the application–level QoS of the inter–destination synchronization schemes, we need to examine the inter–destination synchronization quality as well as the intra–stream and inter–stream synchronization quality.

For the inter–destination synchronization quality, we evaluate the mean square error of inter–destination synchronization5. In two destinations A and B, it denotes the mean square of the difference between the output time of an MU (excluding skipped MUs) at destination A and that of the MU at destination B. In this paper, we suppose many destinations. Thus, there are many combinations of two destinations. However, some combinations have the same tendency as other combinations or very small inter–destination synchronization error. Therefore, in this paper, we select a reference destination from among all the destinations and then calculate the average of mean square errors of inter–destination synchronization between the reference destination and another one; we use it for quality assessment. We have also measured the standard deviation of the mean square errors. However, we found that the standard deviation has the same tendency as that of the average; thus, we do not show the results of the standard deviation.

On the other hand, for the quality assessment of intra–stream synchronization for audio or video, we evaluate the coefficient of variation of output interval, which represents the smoothness of output of a media stream.

We have also assessed inter–stream synchronization quality in the simulation. As a result, we noticed that all the schemes have high inter–stream synchronization quality in the simulation. Thus, we do not show the result.

IV. SIMULATION RESULTS

In this section, we compare the application–level QoS of the five schemes defined in Subsection III-B. Before the comparison, we investigate the relationship between the location of the centralized control terminal and the application–level QoS of the media streams in the synchronization maestro and master–slave destination schemes. In the comparison, we first assess the influence of the amount of the interference traffic. Furthermore, the relationship between the number of destinations and the application–level QoS of the schemes is investigated.

In this paper, each symbol in the figures to be shown represents the average of 10 measured values which were obtained by changing the random seed for generating the interference traffic. We also show 95% confidence intervals of the QoS parameters in the figures. However, when the interval is smaller than the size of the corresponding symbol representing the simulation result, we do not show it in the figures.

A. Location of Maestro and Master

In this subsection, we set x to 6; that is, 11 destinations exist in the network. We change average load 2 from 2.5 Mbps to 3.6 Mbps with average load 1 fixed at 2.5 Mbps.

Figure 2 shows the average of mean square errors of inter–destination synchronization for voice between MR11 and another destination versus average load 2. Figure 3 depicts the coefficient of variation of output interval for voice at MR11 versus average load 2. In these figures, Maestro–MS and Maestro–MRi (i = 10, 11) mean that we choose MS and MRi as the synchronization maestro terminal, respectively. Furthermore, Master–MRj (j = 1, 10, 11) specifies that we select MRj as the master destination.

We first discuss the synchronization maestro scheme. From Fig. 2, we find that Maestro–MR11 has smaller inter–destination synchronization error for voice than Maestro–MS and Maestro–MR10. However, Fig. 3 shows that for average load 2 heavier than about 2.9 Mbps, the coefficient of variation for voice at MR11 with Maestro–MR11 is the largest. The reason is as follows. When we select MR11 as the maestro, the voice and video MUs received by MR11 go through the same links as the control packets sent by each destination to the maestro. Then, under a heavily loaded condition, as the number of control packets increases, the voice and video MUs are lost frequently; this degrades the output quality of the media streams. Thus, MR11 is not appropriate for the maestro.

In Figs. 2 and 3, we also notice that Maestro–MR10 has almost the same media synchronization quality as that with

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>SPECIFICATIONS OF THE VOICE AND VIDEO.</th>
</tr>
</thead>
<tbody>
<tr>
<td>item</td>
<td>voice</td>
</tr>
<tr>
<td>coding scheme</td>
<td>ITU–T</td>
</tr>
<tr>
<td>image size [pixels]</td>
<td>G.711 µ-law</td>
</tr>
<tr>
<td>original average MU size [bytes]</td>
<td>400</td>
</tr>
<tr>
<td>original average MU rate [Mbps]</td>
<td>20.0</td>
</tr>
<tr>
<td>original average inter–MU time [ms]</td>
<td>50.0</td>
</tr>
<tr>
<td>measurement time [s]</td>
<td>64.0</td>
</tr>
</tbody>
</table>
We now compare the application-level QoS of the five schemes defined in Subsection III-B. We use the same values of $x$, average loads 1 and 2 as those in the previous subsection.

**B. QoS Comparison among Inter–Destination Synchronization Schemes**

From the previous results, we choose the source terminal MS as the synchronization maestro in Maestro. We also select the destination terminal MR11, which is the most heavily loaded destination in the simulation, as the master in Master–Slave.

Figures 4 and 5 show the average of mean square errors of inter–destination synchronization between MR11 and another destination for voice and for video, respectively. In Fig. 6, we present the average MU delay of voice at MR11 versus average load 2. The coefficient of variation of output interval for voice at MR11 is shown as a function of average load 2 in Fig. 7. From Figs. 4 and 5, we find that the inter–destination synchronization errors of voice have a tendency similar to those of video. We also notice in Figs. 4 and 5 that the inter–destination synchronization errors of Maestro, Master–Slave and Distributed are smaller than that of VTR. Furthermore, NC has the largest inter–destination synchronization error. Therefore, we can say that the inter–destination synchronization control is effective in improving the inter–destination synchronization quality.

In these figures, we observe that when average load 2 is heavier than about 3.0 Mbps, the average of mean square errors with Distributed is the smallest among all the schemes. This is because the difference in MU delays between destinations becomes small at the expense of MU loss. In the distributed control scheme, each destination transmits many control packets to all the other destinations; accordingly, all the routers may receive many control packets. Every control packet is smaller than the other kinds of packets and then stays on a link for...
shorter duration. In the simulation, the maximum queue length is defined by the number of packets and is not related to the size of each packet\(^6\). Thus, as the percentage of the control packets in a router queue increases, the waiting time in the router queue becomes small. As a result, the transfer delay time of an MU and the difference in the MU transfer delays between destinations decrease. Therefore, the inter–destination synchronization error with Distributed is small. The influence of MU loss will be mentioned later.

Furthermore, in Fig. 5, the average of mean square errors for video with Maestro is equal to or slightly larger than that of Master–Slave. This is because the synchronization maestro is implemented at the source, and the maestro sends the control packets periodically and when the reference output timing is changed. If we select the source as the synchronization maestro, control packets from the synchronization maestro to destinations are transmitted through the same route as that of MUs. Thus, under a heavily loaded condition, control packets as well as voice and video MUs may drop. When the control packet is lost, the destination cannot get the reference output timing until receiving the next control packet. Therefore, the inter–destination synchronization quality with Maestro deteriorates.

On the other hand, in Master–Slave, we have selected MR11 as the master destination; it is the most heavily loaded destination in the simulation. MR11 does not need to receive the control packets from the other destinations. Furthermore, on the links from the router R2\(x\) (i.e., R12) to R2 in Fig. 1, the control packets are transmitted in the opposite direction to the voice and video MUs. Thus, the control packets do not compete with MUs and then rarely drop on the links. Therefore, Master–Slave achieves higher inter–destination synchronization quality than Maestro. However, Master–Slave has a problem that the most heavily loaded destination is not always known. Owing to this, we cannot always select the master destination appropriately.

Figure 6 shows that for average load 2 heavier than about 3.2 Mbps, the average MU delay of voice with Distributed is the smallest among all the schemes. Hence, we can say that in Distributed, the waiting time in each router queue becomes small owing to the control packets.

In Fig. 7, we can confirm that for average load 2 heavier than about 3.1 Mbps, the coefficient of variation of output interval for voice with Distributed is the largest among all the schemes. This is due to the huge amount of control packets; accordingly, voice MUs drop frequently at the router queues. Thereby, intra–stream synchronization quality with Distributed degrades largely. Thus, we can say that this scheme is not appropriate to our network configuration.

C. Influence of the Number of Destinations

In order to assess the influence of the number of destinations on the application–level QoS, we change \(x\) from 3 to 18; that is, the number of destinations varies from 5 to 35. We keep average load 2 at 3.3 Mbps, while average load 1 is set to 2.5 Mbps. We choose the source terminal MS as the synchronization maestro in Maestro and select the destination terminal MR2\(x\) – 1 as the master in Master–Slave.

Figure 8 presents the average of mean square errors of inter–destination synchronization for voice between MR1 and another destination versus the number of destinations. Figure 9 plots the average MU delay of voice at MR5 versus the number of destinations.

In Fig. 8, we notice that when the number of destinations is smaller than 25, the average of mean square errors with Distributed is the smallest among all the schemes, and that with Master–Slave is the second smallest. These results are similar to those in the previous subsection.

This figure also shows that the inter–destination synchronization error with Maestro has a local peak at around 15 destinations. The reason is as follows. As the number of destinations increases, the loss rate of the control packets at the most heavily loaded destination increases; this degrades inter–destination synchronization quality. On the other hand, the reference target output time largely increases as the number of destinations increases. However, owing to the constraint of the maximum allowable delay \(\Delta_{al}\), the reference target output time cannot exceed the limit and therefore saturates; accordingly, the modi-
We also find in Fig. 8 that for the number of destinations larger than 27, the inter-destination synchronization error with Master–Slave and that with Maestro increase largely as the number of destinations increases. This is because the output time of voice MUs at $MR2x - 1$ exceeds the target output time which is limited by the maximum allowable delay $\Delta_{al}$. Thereby, the maestro or master cannot set the reference output timing to appropriate one for $MR2x - 1$, and then the inter-destination synchronization error increases as the number of destinations increases. We can notice it in Fig. 9, in which the average MU delay with Maestro and that with Master–Slave take an approximately constant value of 300 ms when there are more than 25 destinations.

We have also investigated the coefficient of variation of output interval for voice versus the number of destinations, which is not shown here because of limitations of space. As a result, we noticed that the coefficient of variation with Distributed increases as the number of destinations increases and is much larger than that with Maestro and Master–Slave. This is because the number of control packets increases as the number of destinations increases in Distributed.

We can summarize the above results as follows. When the network traffic condition is not known, we should use Maestro. On the other hand, if we can get the network traffic condition exactly, Master–Slave is the best choice. However, the most appropriate scheme depends on the acceptable inter-destination synchronization quality of applications. If we use the application which is severe with inter-destination synchronization quality, we should employ Distributed. However, it degrades the intra-stream synchronization quality at heavily loaded destinations.

V. CONCLUSIONS

In this paper, we compared the application-level QoS of the three inter-destination synchronization schemes: the master-slave destination scheme, the synchronization maestro scheme, and the distributed control scheme. We then noticed that the inter-destination synchronization quality with the distributed control scheme is higher than the other schemes, although the intra-stream synchronization quality with the distributed control scheme under a heavily loaded condition is largely degraded. We also found that the inter-destination synchronization quality with the master-slave destination scheme is higher than that with the synchronization maestro scheme when we select the most heavily loaded destination as the master; however, the master-slave destination scheme has a problem that the most heavily loaded destination is not always known, and we cannot always select the master destination appropriately.

Therefore, we should use the synchronization maestro scheme if it is not clear which destination is the most heavily loaded. On the other hand, when it is known, the master-slave destination scheme is the best choice. However, if the application demands very high inter-destination synchronization quality, we should use the distributed control scheme.

As the next step of our research, we plan to assess the user-level QoS of the inter-destination synchronization schemes. We also need to investigate the relationship between the user-level QoS and the application-level QoS. In addition, the QoS comparison in other network configurations is needed. Furthermore, it is a future study that we employ a network management protocol for knowing the network state in order to choose appropriate inter-destination synchronization schemes.


ded by the maximum allowable delay $\Delta_{al}$. Thereby, the maestro or master cannot set the reference output timing to appropriate one for $MR2x - 1$, and then the inter-destination synchronization error increases as the number of destinations increases. We can notice it in Fig. 9, in which the average MU delay with Maestro and that with Master–Slave take an approximately constant value of 300 ms when there are more than 25 destinations.

We have also investigated the coefficient of variation of output interval for voice versus the number of destinations, which is not shown here because of limitations of space. As a result, we noticed that the coefficient of variation with Distributed increases as the number of destinations increases and is much larger than that with Maestro and Master–Slave. This is because the number of control packets increases as the number of destinations increases in Distributed.

We can summarize the above results as follows. When the network traffic condition is not known, we should use Maestro. On the other hand, if we can get the network traffic condition exactly, Master–Slave is the best choice. However, the most appropriate scheme depends on the acceptable inter-destination synchronization quality of applications. If we use the application which is severe with inter-destination synchronization quality, we should employ Distributed. However, it degrades the intra-stream synchronization quality at heavily loaded destinations.

V. CONCLUSIONS

In this paper, we compared the application-level QoS of the three inter-destination synchronization schemes: the master-slave destination scheme, the synchronization maestro scheme, and the distributed control scheme. We then noticed that the inter-destination synchronization quality with the distributed control scheme is higher than the other schemes, although the intra-stream synchronization quality with the distributed control scheme under a heavily loaded condition is largely degraded. We also found that the inter-destination synchronization quality with the master-slave destination scheme is higher than that with the synchronization maestro scheme when we select the most heavily loaded destination as the master; however, the master-slave destination scheme has a problem that the most

\[ \text{Average MU delay of voice} \quad \text{(influence of the number of destinations)} \]

![Fig. 9. Average MU delay of voice at MR5](image)

\[ \text{Average load 1 = 2.5 Mbps} \quad \text{Average load 2 = 3.3 Mbps} \]

\[ \text{95\% confidence interval} \]

\[ \text{Number of destinations} \]

\[ \text{Average MU delay of voice} \quad [\text{ms}] \]

\[ \text{Average load 1 = 2.5 Mbps} \quad \text{Average load 2 = 3.3 Mbps} \]

\[ \text{95\% confidence interval} \]

\[ \text{Number of destinations} \]

\[ \text{Average MU delay of voice} \quad [\text{ms}] \]

\[ \text{Average load 1 = 2.5 Mbps} \quad \text{Average load 2 = 3.3 Mbps} \]

\[ \text{95\% confidence interval} \]

\[ \text{Number of destinations} \]

\[ \text{Average MU delay of voice} \quad [\text{ms}] \]

\[ \text{Average load 1 = 2.5 Mbps} \quad \text{Average load 2 = 3.3 Mbps} \]

\[ \text{95\% confidence interval} \]

\[ \text{Number of destinations} \]

\[ \text{Average MU delay of voice} \quad [\text{ms}] \]

\[ \text{Average load 1 = 2.5 Mbps} \quad \text{Average load 2 = 3.3 Mbps} \]

\[ \text{95\% confidence interval} \]

\[ \text{Number of destinations} \]

\[ \text{Average MU delay of voice} \quad [\text{ms}] \]

\[ \text{Average load 1 = 2.5 Mbps} \quad \text{Average load 2 = 3.3 Mbps} \]

\[ \text{95\% confidence interval} \]

\[ \text{Number of destinations} \]

\[ \text{Average MU delay of voice} \quad [\text{ms}] \]

\[ \text{Average load 1 = 2.5 Mbps} \quad \text{Average load 2 = 3.3 Mbps} \]

\[ \text{95\% confidence interval} \]

\[ \text{Number of destinations} \]